Lund, 6 December 2011

CasADi tutorial – Introduction

Joel Andersson   Johan Åkesson

Department of Electrical Engineering (ESAT-SCD) &
Optimization in Engineering Center (OPTEC)
Katholieke Universiteit Leuven

OPTEC – Optimization in Engineering

- Interdiciplinary center at Katholieke Universiteit Leuven, Belgium (Flemish region) since 2005
- 20 professors, 10 postdocs and 60+ PhD students from Mech.Eng, Elec.Eng, Civ.Eng, Comp.Sc.
- Principle investigator: Moritz Diehl
- Aim: Bridge the gap between state-of-the optimization algorithms and applications
- Applications: Kite power, river networks, robot control, chemical reactors, etc.

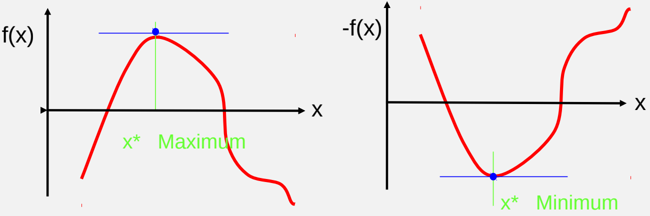
www.kuleuven.be/optec

Outline

What is optimization?

- Optimization = search for the best solution
- in mathematical terms:
  minimization or maximization of an objective function $f(x)$
  depending on variables x subject to constraints

Equivalence of maximization and minimization problems:
(from now on only minimization)



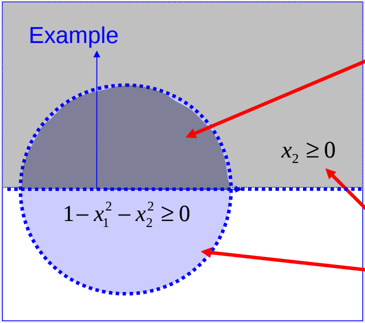
Constrained optimization

- Often variable x shall satisfy certain constraints, e.g.: $x \geq 0$, $x_1^2 + x_2^2 = C$.
- General formulation

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g(x) \geq 0, \quad h(x) = 0 \end{array} \tag{1}$$

- f objective (cost) function
- g inequality constraint function
- h equality constraint function

Feasible set = collection of all points that satisfy all constraints:



feasible set is intersection of grey and blue area

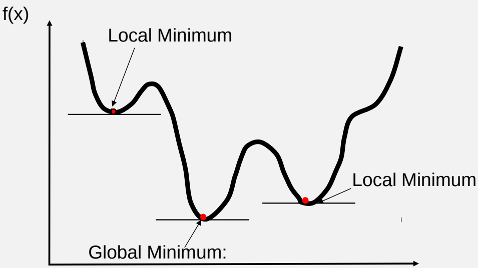
$h_1(x) := \quad x_2 \geq 0$

$h_2(x) := \quad 1 - x_1^2 - x_2^2 \geq 0$

The "feasible set" Ω is $\{x \in \mathbb{R}^n | g(x) = 0,\ h(x) \geq 0\}$.

Local and global minima

- Global minimizer: $x^* \in \Omega$ and $\forall x \in \Omega : f(x) \geq f(x^*)$
- Local minimizer: $x^* \in \Omega$ and there exists a neighborhood $\mathcal{N}$ of x^* such that $\forall x \in \Omega \cap \mathcal{N} : f(x) \geq f(x^*)$



Derivatives

- First and second derivatives of the objective function or the constraints play an important role in optimization
- The first order derivatives are called the gradient

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]^{\mathsf{T}} \tag{2}$$

- When f is vector valued we have the Jacobian: $J(x) = (\nabla f(x))^{\mathsf{T}}$
- The Hessian matrix contains second order derivatives:

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \, \partial x_2} & \ldots & \frac{\partial^2 f}{\partial x_1 \, \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \, \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \ldots & \frac{\partial^2 f}{\partial x_n \, \partial x_n} \\ \ldots & \ldots & \ldots & \ldots \\ \frac{\partial^2 f}{\partial x_n \, \partial x_1} & \frac{\partial^2 f}{\partial x_n \, \partial x_2} & \ldots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{3}$$

Optimization – an overview

Methods to calculate derivatives

- Finite differences
  - Low accuracy, relatively slow
- Symbolic differentiation (in e.g. Maple)
  - Only for small expressions
- Automatic/algorithmic differentiation (AD)
  - Fast & accurate
  - Tools: ADOL-C, CppAD, (CasADi)

Automatic/algorithmic differentiation (AD)

Given a function $f : \mathbf{R}^N \to \mathbf{R}^M$, AD cheaply delivers:

- Forward directional derivatives:

$$y_{\text{fsens}} = \frac{\partial f}{\partial x} x_{\text{fseed}} \tag{4}$$

- Adjoint directional derivatives:

$$x_{\text{asens}} = \left(\frac{\partial f}{\partial x}\right)^{\mathsf{T}} y_{\text{aseed}} \tag{5}$$

- Complete Jacobians/Hessians (cheap if sparse):

$$J = \frac{\partial f}{\partial x} \tag{6}$$

Optimization – an overview

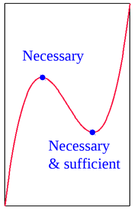
Optimality conditions

- Unconstrained problem:

$$\text{minimize} \quad f(x), \quad x \in \mathbb{R}^n \tag{7}$$

  Assume that f is twice differentiable. We want to test a point x^* for local optimality

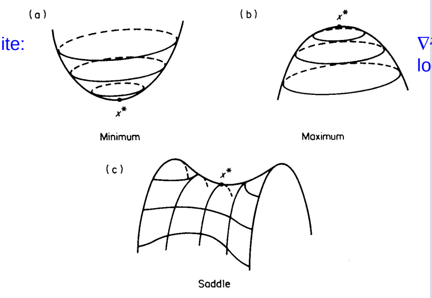
- Necessary condition $\nabla f(x^*) = 0$ (stationarity)
- Sufficient condition x^* stationary and $\nabla^2 f(x^*)$ positive definite



Stationary points

(a)-(c) x^* is stationary: $\nabla f(x^*)=0$

$\nabla^2 f(x^*)$ positive definite: local minimum



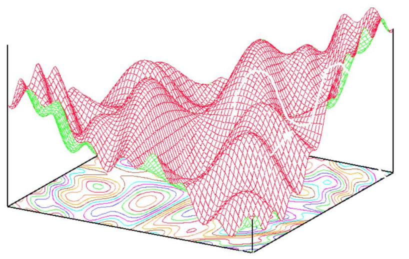
$\nabla^2 f(x^*)$ negative definite: local maximum

$\nabla^2 f(x^*)$ indefinite: saddle point

Optimization – an overview

Local and global optimization

- Sometimes there are many local minima



- Global optimization is a very hard issue - most algorithms find only the next local minimum.
    - Exception: Convex optimization problems

Classes of optimal control problems

- Linear programming (LP)

$$\begin{array}{ll} \text{minimize} & c^{\mathsf{T}}\,x \\ \text{subject to} & x_{\text{min}} \leq x \leq x_{\text{max}} \\ & g_{\text{min}} \leq A\,x \leq g_{\text{max}} \end{array} \tag{8}$$

- Quadratic programming (QP)

$$\begin{array}{ll} \text{minimize} & x^{\mathsf{T}}\,H\,x + c^{\mathsf{T}}\,x \\ \text{subject to} & x_{\text{min}} \leq x \leq x_{\text{max}} \\ & g_{\text{min}} \leq A\,x \leq g_{\text{max}} \end{array} \tag{9}$$

  - Convex QP $\Rightarrow$ H is positive definite
  - Tools: quadprog, CPLEX, OOQP, qpOASES, ...

Nonlinear programming (NLP)

$$
\begin{array}{ll}
\text{minimize} & f(x) \\
\text{subject to} & x_{\text{min}} \le x \le x_{\text{max}} \\
 & g_{\text{min}} \le g(x) \le g_{\text{max}}
\end{array}
\tag{10}
$$

- Nonconvex in general $\Rightarrow$ initial guess important
- Tools exist handling millions of variables & constraints

Solution methods for NLP

- *Sequential quadratic programming* (SQP)
  - At solution guess, quadratic approximation of f and $g \Rightarrow$ QP
  - Solve QP to get a new a new solution guess
  - Continue until necessary conditions for optimality satisfied
  - Tools: fmincon, SNOPT, KNITRO, ...
- *Interior point* or *barrier* methods (IP)
  - Relaxation: penalize inequality constraint violation
  - Solve equality constrained problem with Newton's method with increasing barrier
  - Continue until necessary conditions for optimality satisfied, barrier infinite
  - Tools: IPOPT, KNITRO, ...
- Require first and (preferably) second order derivatives
- Exercise tomorrow

Optimal control problem (OCP) (dynamic optimization)

$$
\begin{array}{ll}
\text{minimize} & \displaystyle\int_{t=0}^{T} L(x, u, p), dt + E(x(T), p) \\
\text{subject to} & \\
& \dot{x}(t) = f(x(t), u(t), p), \qquad t \in [0, T] \\
& x(0) = x_0(p) \\
& x_{\min} \leq x(t) \leq x_{\max}, \qquad t \in [0, T] \\
& u_{\min} \leq u(t) \leq u_{\max}, \qquad t \in [0, T] \\
& p_{\min} \leq p \leq p_{\max}
\end{array}
\tag{11}
$$

- x: *state*, u: *control*, p: *free parameters*
- $\dot{x}(t) = f(x(t), u(t), p)$, $x(0) = x_0(p)$: *system dynamics*
- $L(x, u, p)$: *Lagrange term*, $E(x, p)$ *Meyer term*

More general OCPs

- Multiple dynamic stages
- Path constraints
- Differential algebraic equations (DAE) instead of ODE
- Explicit time dependence
- Multipoint constraints: $r(x(t_0), x(t_0), \ldots, x(t_{\text{end}})) = 0$

Solution methods for OCP

- Hamilton-Jacobi-Bellmann equation / Dynamic Programming
  - Finds *global* optimum through smart exhaustive search
- Indirect Methods
  - Solve necessary conditions for optimality (Pontryagin's Maximum Principle)
- Direct Methods (control discretization)
  - Reformulate OCP as an NLP

Tools for OCP

- JModelica.org, ACADO Toolkit, MUSCOD-II, DyOS, DIRCOL, GPOPS, ...

CasADi

What is CasADi?

- An open-source (LGPL) symbolic framework for quick, yet efficient, implementation of derivative based algorithms for dynamic optimization
- "Framework for writing OCP solvers"
  - Reformulate OCP $\Rightarrow$ NLP
  - Solve NLP efficiently
- Used in JModelica.org

www.casadi.org

What is CasADi?

Why write your own OCP solver?

- Learn about the methods
- Implement new OCP-algorithms
- Treat arbitrary complex OCP
- Full control if something goes wrong

Alternative: Algebraic modelling languages

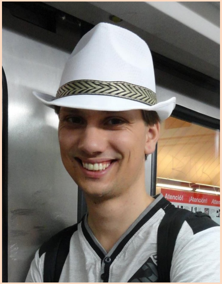
- Tools: AMPL, GAMS, Pyomo, OpenOpt
- High level formulation
- Less flexibility
- Only for collocation
- (Mostly) commercial

What is CasADi?

Main components of CasADi

- A minimalistic Computer Algebra System (CAS) (cf. Symbolic Toolbox for Matlab)
- 8 flavors of automatic differentiation (AD)
  - Forward and adjoint mode
  - Symbolic and numeric
  - Two different ways to represent expressions
- Interfaces
  - Ipopt, Sundials, (KNITRO, OOQP, qpOASES, CPLEX, LAPACK, CSparse, ACADO Toolkit. . .)
- Front-ends
  - C++, Python, (Octave)
- Model import from JModelica.org

Main developers



Myself

Joris Gillis

Contributing

Carlo Savorgnan, Attila Kozma, Greg Horn, Johan Åkesson & Co.

Outline

1. Optimization – an overview
2. What is CasADi?
3. Schedule of tutorials
4. Exercise: Getting started

Schedule of tutorials

Tuesday 6 December

13.15 – 14.00	Introduction
14.00 – 17.00	Exercise: Getting started with CasADi
	Extra exercise: ODE/DAE sensitivity analysis

Wednesday 7 December

8.15 – 12.00	Exercise: NLP

Thursday 8 December

8.15 – 9.00	Optimal control methods
9.00 – 9.30	Advanced concepts in CasADi
9.30 – 12.00	Exercise: Matrix symbolics & OCP with CasADi
13.15 – 14.00	JModelica.org introduction
14.00 – 17.00	Exercise: OCP with JModelica.org

Outline

1. Optimization – an overview
2. What is CasADi?
3. Schedule of tutorials
4. Exercise: Getting started

Exercise: Getting started

Fundamental types

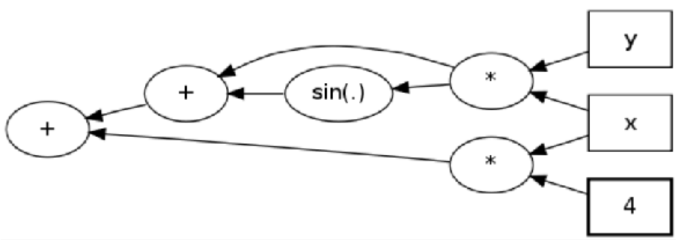
- `SX` – scalar symbolic type
- `SXMatrix` and `DMatrix` – sparse matrices
- `FX` and derived classes – CasADi functions
- `MX` – matrix symbolic type $\Rightarrow$ Thursday morning

Exercise: Getting started

SX

Represents symbolic expressions made up by a sequence of unary and binary operations

- Scalar type
- Used like symbolic variables in `SymPy`/`Maple`/Symbolic Toolbox/...
- Representation like types in AD-tools (cf. `adouble` in ADOL-C)
- Example: $x\,y + \sin(x\,y) + 4\,x$



Exercise: Getting started

`SXMatrix`

- Sparse matrix containing `SX` scalars
- "Everything is a matrix" syntax
- Use this class instead of `SX` directly!

`DMatrix`

- Sparse matrix containing `double` (= `float` in Python)
- Used internally

`FX` and derived classes

- Base class for different kind of *functions*
    - Defined by symbolic expressions
    - ODE/DAE integrators
    - Much more
- Defines a standard way of communicating with these functions
    - Setting options
    - Passing inputs
    - Evaluation
    - Directional derivatives
    - Obtaining the Jacobian

Exercise: Getting started

Exercise take-away

- Working with symbolic expressions
- AD in CasADi

Exercise: Getting started

Extra exercise: ODE/DAE integration & sensitivity analysis

- Forward/adjoint sensitivity analysis in CasADi
- Open-source integrator suite Sundials (ODE: CVodes / DAE: IDAS)
  - Usage: `integrator = CVodesIntegrator(ode_function)`
  - CasADi will formulate the necessary equations
- Sensitivity analysis works just like AD
- How to write your own integrator